FORMULATING EFFECTIVE COORDINATION STRATEGIES IN AGILE GLOBAL SOFTWARE DEVELOPMENT TEAMS

Research-in-Progress

Ye Li
University of Mannheim
Chair of Information Systems IV
Schloss, 68163 Mannheim, Germany
yeli@es.uni-mannheim.de

Alexander Maedche
University of Mannheim
Chair of Information Systems IV
Institute for Enterprise Systems
Schloss, 68131 Mannheim, Germany
maedche@es.uni-mannheim.de

Abstract

Coordination of interdependent tasks across geographical, temporal and socio-cultural boundaries has been identified as a critical antecedent to global software development (GSD) team performance. Since GSD teams are increasingly required to achieve software development agility and respond to changing business requirements, conventional plan-driven software development methods become inadequate to manage the cross-boundary dependencies. To cope with this issue, there is an emerging trend of implementing change-driven agile methods in GSD; we call this practice agile GSD. This paper focuses on the formulation of coordination strategies in agile GSD. We argue that Coordination Theory (CT) can serve as a useful lens to develop a deeper understanding of the formulation process. We present a case study in a leading software development company using agile GSD and present preliminary results by applying the lens of CT.

Keywords: Agile software development, global software development, coordination theory, coordination strategy, coordination mechanism

Introduction

Global software development (GSD) teams consist of people from different locations, normally in different countries, working together to develop and deliver software products (Carmel 1999). In GSD process, a great number of dependencies exist across geographical, temporal and socio-cultural boundaries, since software parts developed in different locations need to be integrated and individual work schedules need to be synchronized (Espinosa et al. 2001). Thus, coordination, defined as managing dependencies among task activities (Malone and Crowston 1994), is highly important for GSD team performance (Chiang and Mookerjee 2004). Although GSD has become common practice in software industry (Holmstrom et al. 2006), effective coordination across the boundaries is still a challenging undertaking (Espinosa et al. 2012).

Much literature has investigated how to manage the cross-boundary dependencies in GSD. In the past, requirements on GSD projects were typically stable or predictable during the development process and the cross-boundary dependencies could be largely managed mechanistically (e.g., through defined procedures, rules, documentation, and detailed plans) (Cataldo et al. 2007; Espinosa and Pickering 2006; Lee et al. 2006). However, as business environments become more and more turbulent and demand rapid technological innovations, GSD teams increasingly strive to develop abilities to swiftly respond to dynamic business requirements (Batra et al. 2010; Ramesh et al. 2006); the abilities are referred to as *software development agility* in the software industry (Lee and Xia 2010). In this circumstance, mechanistic ways of managing dependencies become less effective, since the predefined procedures, rules or plans may become obsolete and require updates soon after created (Batra et al. 2010). To cope with the changing requirements, GSD teams start practicing agile methods (e.g., Scrum, XP) or specific agile techniques (e.g., pair programming, continuous integration) (Batra et al. 2010; Dingsøyr et al. 2012; Hanssen et al. 2011); we term this practice as *agile GSD*.

Agile methods are sets of techniques evolved from personal experiences and collective wisdom in the software community (Abrahamsson et al. 2002). Early studies on agile methods advocated that the application of agile was limited to small and collocated teams (Highsmith 2002; Turk et al. 2002). In recent years, agile methods are more often considered applicable in large teams and distributed settings as well if they are adequately tailored to the given circumstance (Batra et al. 2010; Lee and Yong 2009). Moreover, there is empirical evidence showing that some agile techniques can strengthen communication and team collaboration and thus are more effective in managing cross-boundary dependencies in GSD than techniques in plan-driven methods (e.g., waterfall) (Holmström et al. 2006; Paasivaara and Lassenius 2006; Pikkarainen et al. 2008). However, according to multiple literature reviews (Dingsøyr et al. 2012; Dybå and Dingsøyr 2008; Hanssen et al. 2011; Jalali and Wohlin 2010), there is a lack of theoretical understanding of which agile techniques should be applied and how the agile techniques can be integrated with the plan-driven techniques to achieve a desirable level of coordination.

Coordination theory (CT) has been extensively applied in analyzing coordination processes in teams (Crowston et al. 2006; Faraj and Sproull 2000; Malone and Crowston 1994); it has the potential to serve as a theoretical ground for coordination studies in agile GSD. In the vocabulary of CT, *coordination* is "managing dependencies among task activities; actions taken to manage the dependencies are *coordination mechanisms*; and a set of prioritized coordination mechanisms for a given circumstance is a *coordination strategy*. Teams formulate coordination strategies to manage dependencies and achieve a certain level of *coordination effectiveness*, defined as to which extent dependencies are well managed (Crowston et al. 2006). Some studies leverage CT to analyze coordination processes in conventional GSD teams. They suggest that the availability and effectiveness of coordination mechanisms is influenced by numerous situational factors (e.g., task, team, technology, organization); to achieve a desirable level of coordination effectiveness, teams have to clearly identify critical situational factors and formulate a tailored coordination strategy (Espinosa et al. 2002; Espinosa and Pickering 2006). Analyzed with the framework of CT, agile GSD is substantially different from the conventional GSD in three key aspects: first, the situational factors (e.g., task routineness, team empowerment, customer requirements) are largely different in agile GSD and in conventional GSD (Abrahamsson et al. 2002). Second, there are more coordination mechanisms (e.g., software development techniques) available in agile GSD than in conventional GSD. Last, the impact of coordination effectiveness on team performance is greater in agile

GSD than in conventional GSD. Due to such differences, the findings of coordination studies in conventional GSD have limited generalizability to agile GSD.

The main objective of this paper is to *develop an empirically based understanding of which situational factors in agile GSD influence the formation of coordination strategies, and how and why they exert the influence*. To accomplish the objective, we conducted an interpretive case study in a leading software development company that specifically engaged in agile GSD in the past three years to develop and deliver software products worldwide. We interviewed key coordinators in the agile GSD teams to elicit their critical decisions on the formulation of coordination strategies. Following an inductive approach for data analysis, we seek to identify influential situational factors in the decisions, understand how they affect the formation of coordination strategies, and identify some of the contingencies that may influence the importance of the situational factors or the effectiveness of coordination mechanisms. By doing this, we expect to present a clearer view on the formation of coordination strategies in agile GSD and provide actionable guidance for practitioners to achieve a desirable level of coordination effectiveness.

Theoretical Foundations

Studies on coordination emerged and thrived in various disciplines (e.g., economics, computer science, organization theory) during 1980s. In 1994, Malone and Crowston published an interdisciplinary study on coordination, which created synergy among the different disciplines. In this work, they established a set of interlinked concepts and patterns to analyze coordination and named it Coordination Theory (Malone and Crowston 1994). For the purpose of this study, we find CT particular useful, because it incorporates a socio-technical view of coordination and suggests a set of situational factors that might influence the formulation of coordination strategies (Crowston et al. 2006; Lu et al. 2011). Thus, we use CT as the main theoretical lens. Following Malone and Crowston's (1994) framework, we view agile GSD process as a group of actors (e.g., developers, product owners, customers) performing interdependent tasks (e.g., planning, coding, review, delivery) and the actors use coordination mechanisms to manage dependencies to achieve a certain level of coordination effectiveness (Crowston et al. 2006; Espinosa and Carmel 2004).

When applied in studies of team process and effectiveness, CT is often integrated with the "input-process-outcome" (I-P-O) model (Hackman 1987), which clearly depicts how situational factors (I) affect the formulation of coordination strategies (P) and finally lead to a certain level of coordination effectiveness (O) (Espinosa and Carmel 2004). In the following, we present key components in the original CT, their instantiation in conventional GSD as well as their implications for agile GSD.

Coordination Effectiveness

Coordination effectiveness refers to the extent to which dependencies among task activities are well managed (Crowston et al. 2006). In software development teams, coordination effectiveness is typically assessed from three aspects: 1) technical- i.e., software parts can be integrated without errors; 2) temporal- i.e., all software parts or sub-processes are completed on schedule; 3) process- i.e., established procedures and priorities are understood and followed (Espinosa et al. 2002; Espinosa and Pickering 2006).

Evidenced by on-time and within-budget delivery of software products that meet customer requirements (Espinosa et al. 2012), team performance is a fundamental interest for studies in software development teams; however, high coordination effectiveness does not always lead to high team performance (Espinosa et al. 2002). This can be explained from two perspectives. First, in addition to coordination effectiveness, there may be other antecedents influencing performance (Espinosa et al. 2002). For example, in conventional GSD, a well-coordinated software team that does not respond to changes in customer requirements may end up delivering software that cannot satisfy customers' needs. Second, some dependencies among the task activities may have much greater influence on team performance than others (Espinosa et al. 2002); if the critical dependencies are not properly handled, no matter how well other dependencies are managed, the team performance is still low. For example, software teams may cope with technical and temporal dependencies very well, but do not manage dependencies with customers effectively; they may end up delivering error-free software timely but useless for the customers.

In agile GSD, teams need to achieve a certain level of software development agility, value customer collaboration and respond to changes during the complete development cycle (Abrahamsson et al. 2002). Thus, agile GSD teams are more likely to be aware of changing customer requirements and incorporate them into working software than conventional GSD teams. That is to say, coordination effectiveness has greater predictive power on team performance in agile GSD than in conventional GSD. Since key dependencies on customers can be monitored and managed, as long as an agile GSD team is well coordinated during the development cycle, the team is highly likely to deliver software products on time, within budget and meanwhile meeting the customer requirements. The increased importance of coordination in agile GSD calls for close attention on the formulation of coordination strategies, which makes the study of coordination strategies in agile GSD highly relevant.

Coordination Strategies and Situational Factors

Coordination strategy is a set of prioritized actions to manage dependencies; the actions are called coordination mechanisms in CT (Malone and Crowston 1994). In general, there are three types of coordination mechanisms: mechanistic, organic and cognitive (Espinosa et al. 2002, 2010). *Mechanistic coordination mechanisms* manage dependencies with plans, processes, routines or other practices (e.g., schedules, procedure manuals); they are most effective in managing routine aspects of tasks (Ven et al. 1976). *Organic coordination mechanisms* manage dependencies through communication (e.g., providing feedback, mutual adjustment); they are most suitable when routines change or when tasks have little or no routine aspects (Ven et al. 1976). Mechanistic and organic coordination mechanisms are explicit coordination mechanisms that require purposeful and sensible implementation.

In contrast, *cognitive coordination mechanisms* are deployed by actors unconsciously through shared cognition; they enable the actors to understand and anticipate task states and member behaviors (Espinosa et al. 2002). For example, shared knowledge of key concepts, processes and products (i.e., shared mental models) and knowledge about who knows what and who does what (i.e., transactive memory system) can help teams achieve higher coordination effectiveness (Espinosa et al. 2007; Rico et al. 2008). Accessibility of cognitive coordination mechanisms is determined by the existing level of shared cognition in teams: The more shared cognition a team has, the more cognitive coordination mechanisms can be leveraged. Existing shared cognition in a team is both influenced by and influences the use of explicit coordination mechanisms. For example, a team frequently coordinates through communication and use of common procedures is more likely to develop shared cognition; a team having a high level of shared cognition can coordinate effectively with the implicit coordination mechanism and may not need to coordinate with frequent communication (Espinosa et al. 2002).

Since dependencies can often be managed by more than one coordination mechanism, teams need to decide which mix of coordination mechanisms should be applied in order to achieve a desired level of coordination effectiveness. There is empirical evidence showing that the effectiveness of coordination mechanisms can be varied by different *situational factors*, such as characteristics of tasks (e.g., routineness), of teams (e.g., size, longevity, geographical, temporal, socio-cultural distances, experience), technology (e.g., available ICT, richness) or of organizations (e.g., organizational culture, power distribution) (Espinosa et al. 2002; Espinosa and Pickering 2006). Therefore, to achieve desirable coordination effectiveness, teams have to skillfully tailor a mix of coordination mechanisms that fit into the given situational factors, which is the *formation of coordination strategies* (Espinosa et al. 2002).

Coordination Strategies in Conventional GSD

Since GSD team members are distributed in more than one location, a geographical boundary exists between actors in the GSD setting. The effort to bridge a geographical boundary can be measured by the effort required for one actor to visit the other (e.g., time and cost for traveling, necessity for visiting permits and visas) (Holmstrom et al. 2006). Generally, when the effort to cross a geographical boundary is high, the frequency of face-to-face and spontaneous communication is low (Holmstrom et al. 2006; Ågerfalk et al. 2005), which leads to constrained accessibility to some of the organic coordination mechanisms. The geographical boundary also limits the utilization of some mechanic coordination mechanisms that use real objects (e.g., task boards or Kanban systems on a wall). Consequently, due to the reduced communication and use of common practice, teams with geographical boundary are difficult to develop shared cognition (Oshri et al. 2008), which reduces the cognitive coordination mechanisms. In

order to coordinate effectively across the geographical boundary, GSD teams have to make extra effort compared with collocated software development teams. For example, they may need to invest in information and communication technology (e.g., teleconference, distributed configuration management systems) and in traveling to enable coordination across geographical boundaries (Šmite et al. 2009).

Some GSD teams distribute across multiple time zones or work in different time shifts, creating temporal boundaries between actors. Temporal boundaries reduce the over-lapping time of the actors and thus constrain the use of coordination mechanisms that require real-time interaction (Huang and Trauth 2008; Ågerfalk et al. 2005). For example, teams may have limited chance to coordinate through synchronous communication (e.g., on the phone) and delays may occur in information exchange and problem solving, which severely disturb the coordination process (Cummings et al. 2009; Sarker and Sahay 2004). Temporal boundaries also lead to a lack of awareness and false assumptions among team members (Komi-Sirviö and Tihinen 2005), which constrains the use of cognitive coordination mechanisms as well. The effort to coordination across temporal boundaries is substantially increased by delays, necessity for clarifications and rework (Espinosa and Pickering 2006; Espinosa and Carmel 2004), and adjustments in working hours to increase the over-lapping time (Espinosa and Carmel 2003).

When GSD team members are distributed in different socio-cultural environments, they may experience difficulty in understanding other members' normative practices and value systems, creating socio-cultural boundaries between the actors (Holmstrom et al. 2006; Shachaf 2008). Socio-cultural boundaries capture both the social stratification differences (e.g., social class, gender, political status, and ethnicity) (Giddens and Duneier 1999 pp. 206–207) and cultural differences (e.g., language, behavioral normal, and belief systems)(Schein 1985) among GSD team members. Socio-cultural boundaries significantly influence how people interact with each other, interpret a certain situation and react to it (Dekker et al. 2008; Leidner and Kayworth 2006). For example, if GSD members speak different languages, exchange of information becomes difficult, communication could breakdown, and misunderstandings might occur, which constrains the use of organic coordination mechanisms (Jarvenpaa and Leidner 1999; Tanner 2009). GSD members working in different socio-cultural environments may have different sub-team cultures and follow diverse work practices, which could impede the implementation of common work practices (Hung and Nguyen 2008). Therefore, the use of mechanistic coordination mechanisms is limited. Due to different expertise, knowledge sets, mental models, languages and value systems, shared cognition is more difficult to be developed among socio-culturally heterogeneous members (Oshri et al. 2008; Ren and Argote 2011), which hinders the leverage of cognitive coordination mechanisms. In addition, socio-cultural boundaries have an influence on actors' interpretation and reaction to time. In some cultures, actors prefer carrying out task activities in a sequential manner and adhering to defined schedules; while in other cultures, actors tend to deal with multiple tasks at one time and change schedules easily when necessary (Hall and Hall 1990). The different views of time certainly complicate the coordination process, especially when actors are distributed not only across the socio-cultural boundaries but also across temporal boundaries.

To address the coordination challenges in GSD, prior studies suggest a variety of coordination mechanisms, which can be largely divided into two sets: reducing cross-boundary dependencies or managing them as they are. To reduce cross-boundary dependencies, GSD teams modularize or separate task activities at different sites (e.g., separate design and testing) (Cataldo et al. 2007; Espinosa and Pickering 2006), centralize critical decisions at one site (Cataldo et al. 2007) or use few or proximate sites (near-shoring) (Cataldo et al. 2007; Holmstrom et al. 2006). To manage cross-boundary dependencies as they are, GSD teams use extensive and detailed documentation (Cataldo et al. 2007; Lee et al. 2006), formulate coordination procedures and routines up front (Espinosa and Pickering 2006; Huang and Trauth 2008), define redundant or liaison roles (Espinosa et al. 2007; Lee et al. 2006), and carry out periodic meetings or on-site visits (Cataldo et al. 2007; Lee et al. 2006).

According to our review, conventional GSD mainly adopts plan-driven methods (Boehm and Turner 2003) and uses mechanistic coordination mechanisms (e.g., plans, procedures, job division) to manage cross-boundary dependencies (Bannerman et al. 2012; Hossain 2008). Although organic coordination mechanisms (e.g., mutual adjustment, communication) are also used in some circumstances, they take a minor role in conventional GSD (Hossain 2008). Cognitive coordination mechanisms (e.g., shared cognition) are difficult to be developed or leveraged in conventional GSD (Joy et al. 2012).

Coordination Strategies in Agile GSD

Distinct from plan-driven methods, agile methods are designed to swiftly respond to changing business requirements (Abrahamsson et al. 2002), which are often described as change-driven methods (Boehm and Turner 2003). According to the Agile Manifesto, agile development values individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan (Abrahamsson et al. 2002; Boehm and Turner 2003; Highsmith 2002); on every aspect of the values, agile is contrasting with conventional plan-driven methods. To achieve agile values, several *agile methods* have been designed (e.g., Scrum, XP), each of which consists of a set of prescribed *agile techniques* (e.g., iterative planning, pair programming, continuous integration) (Abrahamsson et al. 2002; Schwaber and Beedle 2002).

Analyzed with CT, every agile technique that is consciously applied to manage some dependencies among task activities can be seen as an explicit coordination mechanism. Different from plan-driven methods, a large part of agile techniques fall into organic coordination mechanisms, since they manage dependencies through mutual adjustment and communication (e.g., daily stand-up meeting, regular adaptation to changing circumstances, pair programming) (Batra et al. 2010; Hole and Moe 2008; Hossain 2008). Further, some agile techniques provide a different set of mechanistic coordination mechanisms (e.g., Kanban systems, product backlogs, burn down charts) compared with the conventional GSD.

Recent empirical studies in agile GSD indicate that certain agile techniques (e.g., continuous integration, pair programming, iterative planning etc.) are more effective in managing cross-boundary dependencies than techniques in conventional plan-driven methods (Bannerman et al. 2012; Jalali and Wohlin 2010). However, according to CT, the effectiveness of coordination mechanisms is influenced by numerous situational factors (Espinosa et al. 2002) and it is hardly true that a specific coordination mechanism can have the same level of effectiveness in any situation. The suspicion arising from theoretical examination is also confirmed by some recent studies (Hanssen et al. 2011; Pikkarainen et al. 2008; Ågerfalk et al. 2009); although these studies did not examine effects of situational factors on the selection of proper coordination mechanisms directly, they suggested analyzing the effects in future work. Our study tackles this knowledge gap and targets to examine which situational factors influence and how they influence the formulation of effective coordination strategies in agile GSD.

A Preliminary Understanding of Coordination Strategies in Agile GSD

We set up a preliminary framework of coordination strategies in agile GSD based on our prior theoretical examination and literature review. As shown in Figure 1, the left part contains a set of situational factors suggested by literature on the original CT and on conventional GSD (Espinosa et al. 2002; Espinosa and Pickering 2006). This study is going to examine which factors in the given set has a significant influence on the formulation of coordination strategies in agile GSD, and meanwhile to discover which critical factors exist in an agile environment beyond the retrieved set. The middle part of the framework presents the component of a coordination strategy: Available coordination mechanisms. In agile GSD, both agile methods and plan-driven methods serve as available explicit coordination mechanisms, classified as either organic or mechanistic coordination mechanisms; they are influenced by, and influence the availability and size of implicit coordination mechanisms (i.e., cognitive coordination mechanisms). The right part is the outcome of coordination strategies. Since effectiveness of different coordination mechanisms may be varied by the situational factors, teams need to compare different coordination mechanisms and formulate an effective coordination strategy. Then, the coordination strategy will act on dependencies and lead to a certain level of coordination effectiveness. Additionally, agile GSD as an instantiation of agile methods should achieve a certain level of agility and allow changes in requirements during the complete development cycle (Abrahamsson et al. 2002). Some agile techniques support early and frequent assessments of coordination effectiveness (e.g., using burn-down charts to identify temporal coordination problems instantly; continuous integration identifies technical coordination problems promptly) (Abrahamsson et al. 2002). Therefore, whenever an unsatisfactory level of coordination effectiveness is identified, a reexamination on situational factors and consequently an update on coordination strategies should be carried out.

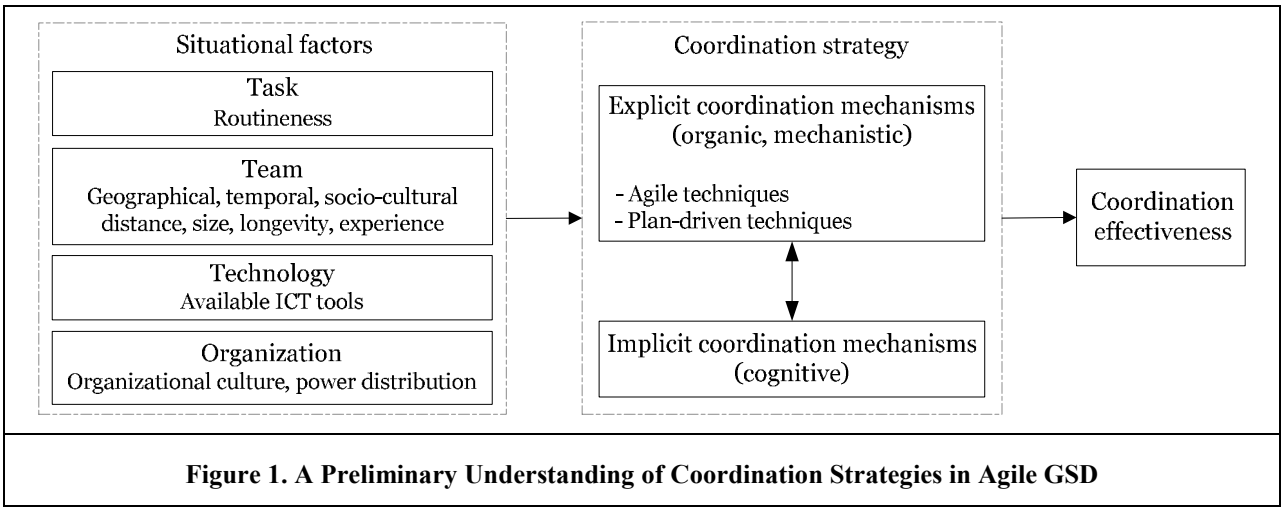

Figure 1. A Preliminary Understanding of Coordination Strategies in Agile GSD

We would like to clarify that the purpose of our study is to examine which situational factors influence and how they influence the formation of effective coordination strategies with evidence emerged from the data; we do not intend to deductively test the preliminary framework in Figure 1. This section should be viewed as a reflection of our theoretical sensitivity. The preliminary framework serves as a legitimate framework containing potentially useful vocabulary to guide our data analysis.

Research Methodology

Research Context

To examine the formulation of coordination strategies in agile GSD, we conducted our study in a leading software development company (GlobCo) that develops and delivers software worldwide. GlobCo values people empowerment and believes being flexible and responsive to business requirements is a key to staying ahead of the competition. Driven by the organizational values, in 2008 GlobCo started to change their plan-driven development methods (e.g., waterfall) into agile methods in existing GSD teams, and gradually stabilized and extended the implementation of agile methods to more development work. By the end of 2011, 12,000 employees had been involved in the agile development worldwide, covering a large variety of software products (e.g., warehouse, business objective software, mobile applications). The Scrum development method is chosen as the main agile method in GlobCo. Although senior management has general recommendations and regulations on which sub-sets of agile techniques in Scrum should be applied, teams are empowered to selectively implement specific techniques among the recommended sets; they can also change or adjust the implemented techniques according to their own situations. Therefore, we have the opportunity to observe and compare implementation status in different agile GSD teams.

We carried out data collection in seven program teams of one large division in GlobCo. Each program team has a unique product scope and is composed of 2 to 10 Scrum teams distributed in four countries in Europe and Asia and each Scrum team consist of 4 to 14 team members; in total, the program teams involve 44 Scrum teams and 421 members (see Table 1). Each program team is led by one program lead (PL) and a chief product owner (CPO); the program lead is responsible for deploying resources, monitoring performance and maintaining a smooth daily-operation, while the chief product owner is the single point of accountability for the success or failure of the program. If multiple Scrum teams of one program are located in the same geographical area, an area product owner (APO) is nominated, who is responsible for the area. Finally, each Scrum team has one team product owner (TPO), ensuring that the team delivers value to the business. For each program team, the PL, the CPO, APOs and TPOs constitute a product owner team. The overarching coordination strategy in the program teams is to locate people in one Scrum team at one site as far as possible, and coordinate the globally distributed Scrum teams through the product owner team to jointly deliver an integrated product. This model is named as "distributed Scrum of Scrum" in Sutherland et al.'s classification (2007), which is recommended as the best practice when Scrum is implemented in GSD teams.

We selected the program team as the unit of analysis; the reasons are twofold: First, program teams are relatively large, which provide a fair sample of subjects and a wide variety of perspectives and stories about the formulation of coordination strategies. Second, the program teams are project-driven and highly focused, and they generally have a stable membership during a release cycle (i.e., six months or one year); therefore, members in one program team have the potential to develop a sense of identity with the team. Finally, different program teams have diverse product scopes and team characteristics (e.g., geographic, temporal and socio-cultural distances, team experience, and team size); therefore, the effects of situational factors on the formulation of coordination strategies can be observed and compared. Although there is not much variation on available technology across product teams, the teams do have full flexibility in determining which technology to use for coordination and the selected technology may exert influences on the formulation of coordination strategies. Since we collect data from one organization, the effects of the organizational environment as a situation factor on the formulation of coordination strategies might not be discovered. However, prior studies suggest that software companies that attempt to achieve software development agility and favor agile methods share some common aspects in their organizational cultures (e.g., value teamwork and social interaction, encourage empowerment of people)(Chan and Thong 2009; Iivari and Iivari 2011; Strode et al. 2009). It is likely that software companies adopting agile GSD have similar organizational environments and thus the organizational factor exerts relatively constant effects on the formulation of coordination strategies.

Data Collection and Data Analysis

The vice president of the operation department in the division provided the authors with entry for conducting fieldwork; thus, the legitimacy and credibility to the researchers was ensured (Patton 1990 p. 254). The vice president introduced the authors to the program teams and a staff member in the operation department assisted in arranging formal face-to-face interviews and telephone interviews. The authors were also provided access to relevant internal documents (e.g., meeting minutes, training materials, internal wiki pages) and allowed to attend team meetings (e.g., product-owner-team meetings, planning meetings and review meetings). The study was initiated in January 2012, and data collection was mainly conducted over a period of five months (February 2012-June 2012).

Table 1. Profiles of Program Teams and Interviewee Distribution

Program team	# of Scrum teams	Program size (Heads)	# of Interviewees (location and roles*)			Total # of interviewees
			Europe	Asia	Rotation**	
Prog#1	6	54	1 PL		1 CPO	2
Prog#2	5	47	2 TPOs			2
Prog#3	8	80	1 TPO	2 TPOs		3
Prog#4	10	100	1 PL, 2 APOs	1 APO, 2 TPOs	1 CA	7
Prog#5	2	12	1 TPO			1
Prog#6	7	69			1 PL, 1 TPO	2
Prog#7	6	59	1 APO	2 TPOs		3
Operation	-	-	1 VP			1
Total	*44*	*421*	*10*	*7*	*4*	*21*

* APO: area product owner; CA: chief architect; CPO: chief product owner; PL: program lead; TPO: team product owner; VP: vice president

** Some interviewees were in a job rotation in Europe when interviewed.

We sought to select interviewees who carry out coordination activities in different program teams, different levels and different sites. In total, we interviewed twenty-one coordinators with diverse roles in seven program teams (see Table 1). The interviews typically lasted 30-50 minutes, all of which were taped and transcribed. We conducted interviews until we reached theoretical saturation (i.e., the last few

interviews did not add new concepts or insights), making the sample size appropriate for this study. The critical incident technique (CIT) (Flanagan 1954) was taken as the interview technique, to intensively examine critical decisions on the formulation of coordination strategies and elicit rich retrospective data from the interviewees. CIT has been successfully applied in various information systems studies (Majchrzak et al. 2005; Thomas and Bostrom 2010). A semi-structured interview protocol was used to guide the interviewees through their experiences of specific critical incidents (i.e., decision points of the formulation of coordination strategies), and the interview questions were designed to proceed from general to specific incidents in order to trace memory. In addition, we also collected data through non-participant observation and archive analysis (Lillis 1999). For each program team, the first author attended one coordination meeting lasting one to three hours (e.g., product-owner-team meeting, Sprint planning meeting or Sprint review meeting) and took field notes on the coordination process. Each program team maintained an internal wiki page, where the minutes of product-owner-team meetings, planning meetings and review meetings were documented. We reviewed the program teams' internal wiki pages and extracted historical data on coordination strategies and successfulness of prior releases (e.g., whether delivered on time, on budget, with targeted functionalities). As discussed in the theoretical foundations, coordination effectiveness has a strong positive correlation with team performance in agile GSD; thus, the successfulness of prior releases as a measurement of team performance can be taken as an indicator of prior coordination effectiveness.

Since the theoretical understanding of the research topic is at an early stage, we predominantly followed an inductive approach for data analysis (Walsham 2006). Sarker and Sarker's (2009, pp. 445-446) interpretive case study guidelines were followed to examine and make sense of the data. Specifically, we identified and refined decision schemes of formulating coordination strategies in agile GSD by "constant comparative analysis". *In-vivo* codes as well as concepts from theories were used in the coding process and refined iteratively. As part of the constant comparison process, whenever possible, we strived to ensure that emerging decision schemes were suggested by multiple interviewees. Observational data collected in the team meetings and historical data extracted from the internal team wiki pages served as snapshots of coordination strategies; they either validated or helped to interpret the interview results. When disagreements among different data sources were detected, we undertook further investigation and sought to discover additional contingencies (Sarker and Sarker 2009).

Preliminary Results and Expected Contributions

In the preliminary analysis of the interview data, we see several opportunities that will contribute to the current knowledge of coordination in agile GSD. First, most situational factors identified in conventional GSD also affect the formulation of coordination strategies in agile GSD, however with different intensities. For instance, a large temporal distance among the actors may increase the use of mechanistic coordination mechanisms (e.g., "You have to go for the backlog structure approach, because the time zone is such a problem that you will hardly find any slot to discuss"). Second, some critical situational factors seldom discussed in conventional GSD emerged from the interviews. For example, changes in customer requirements become an important factor and require both mechanistic and organic coordination (e.g., "also involving the development team with customer calls, we used to have regular customer calls, so we could debrief pretty fast"). Third, we observe that the use of certain explicit coordination mechanisms significantly altered some of the situational factors as well as the cognitive coordination mechanisms over time. For example, several interviewees mentioned that although some of the members had worked together for years in conventional GSD teams, the intensified communication and use of common practices in agile GSD largely improved their mutual trust and shared cognition.

Based on further data analysis, we hope to contribute an in-depth understanding of the formulation of coordination strategies in agile GSD. Future research can leverage the research findings to consciously improve the formulation of coordination strategies and hence improve the coordination effectiveness. In addition, we expect to contribute to the practice of agile GSD by presenting effective ways to formulate coordination strategies.

References

Abrahamsson, P., Salo, O., Ronkainen, J., and Warsta, J. 2002. *Agile Software Development Methods: Review and Analysis*, *VTT Publications 478*.

Ågerfalk, P. J., Fitzgerald, B., Holmström, H., Lings, B., Lundell, B., and Conchúir, E. Ó. 2005. "A Framework for Considering Opportunities and Threats in Distributed Software Development," In *Proceedings of the International Workshop on Distributed Software Development*, pp. 47–61.

Ågerfalk, P. J., Fitzgerald, B., and Slaughter, S. A. 2009. "Introduction to the Special Issue-Flexible and Distributed Information Systems Development: State of the Art and Research Challenges," *Information Systems Research* (20:3), pp. 317–328.

Bannerman, P. L., Hossain, E., and Jeffery, R. 2012. "Scrum Practice Mitigation of Global Software Development Coordination Challenges: A Distinctive Advantage?," In *HICSS 2012 Proceedings*, pp. 5309–5318.

Batra, D., Xia, W., VanderMeer, D., and Dutta, K. 2010. "Balancing Agile and Structured Development Approaches to Successfully Manage Large Distributed Software Projects: A Case Study from the Cruise Line," *Communications of the AIS* (27:1), pp. 379–394.

Boehm, B., and Turner, R. 2003. *Balancing Agility and Discipline: A Guide for the Perplexed*, Boston, MA: Pearson Education, Inc.

Carmel, E. 1999. *Global Software Teams: Collaborating across Borders and Time Zones*, NJ, USA: Prentice Hall PTR Upper Saddle River.

Cataldo, M., Bass, M., Herbsleb, J. D., and Bass, L. 2007. "On Coordination Mechanisms in Global Software Development," In *IEEE International Conference on Global Software Engineering (ICGSE'07)*, pp. 71–80.

Chan, F. K. Y., and Thong, J. Y. L. 2009. "Acceptance of Agile Methodologies: A Critical Review and Conceptual Framework," *Decision Support Systems* (46:4), pp. 803–814.

Chiang, I., and Mookerjee, V. S. 2004. "Improving Software Team Productivity," *Communications of the ACM* (47:5), pp. 89–93.

Crowston, K., Rubleske, J., and Howison, J. 2006. "Coordination Theory," In *Human-Computer Interaction and Management Information Systems: Foundations. Advances in Management Information Systems*, P. Zhang and D. Galletta (eds.), (Vol. I)Armonk, NY: M.E. Sharpe.

Cummings, J. N., Espinosa, J. A., and Pickering, C. K. 2009. "Crossing Spatial and Temporal Boundaries in Globally Distributed Projects: A Relational Model of Coordination Delay," *Information Systems Research* (20:3), pp. 420–439.

Dekker, D. M., Rutte, C. G., and Van den Berg, P. T. 2008. "Cultural Differences in the Perception of Critical Interaction Behaviors in Global Virtual Teams," *International Journal of Intercultural Relations* (32:5), pp. 441–452.

Dingsøyr, T., Sridhar, S., Balijepally, V., and Moe, N. B. 2012. "A Decade of Agile Methodologies: Towards Explaining Agile Software Development," *Journal of Systems and Software* (85), pp. 1213–1221.

Dybå, T., and Dingsøyr, T. 2008. "Empirical Studies of Agile Software Development: A Systematic Review," *Information and Software Technology* (50:9-10), pp. 833–859.

Espinosa, A., Lerch, F., and Kraut, R. 2002. "Explicit vs. Implicit Coordination Mechanisms and Task Dependencies: One Size Does Not Fit All," In *Team Cognition: Process and Performance at the Inter- and Intra-individual Level*, E. Salas, S. M. Fiore, and J. A. Cannon-Bowers (eds.), Washington, DC: American Psychological Association.

Espinosa, J. A., and Carmel, E. 2003. "The Impact of Time Separation on Coordination in Global Software Teams: a Conceptual Foundation," *Software Process: Improvement and Practice* (8:4), pp. 249–266.

Espinosa, J. A., Cummings, J. N., and Pickering, C. 2012. "Time Separation, Coordination, and Performance in Technical Teams," *IEEE Transactions on Engineering Management* (59:1), pp. 91–103.

Espinosa, J. A., and Pickering, C. 2006. "The Effect of Time Separation on Coordination Processes and Outcomes: A Case Study," In *HICSS 2006 Proceedings*, pp. 191–209.

Espinosa, J., Armour, F., and Boh, W. F. 2010. "Coordination in enterprise architecting: an interview study," In *HICSS 2010 Proceedings*, pp. 1–10.

Espinosa, J., and Carmel, E. 2004. "The Effect of Time Separation on Coordination Costs in Global Software Teams: A Dyad Model," In *HICSS 2004 Proceedings*, pp. 1–10.

Espinosa, J., Kraut, R., Lerch, J., Slaughter, S. A., Herbsleb, J. D., and Mockus, A. 2001. "Shared Mental Models and Coordination in Large-Scale, Distributed Software Development," In *ICIS 2001 Proceedings*. Paper 64.
Espinosa, J., Slaughter, S., Kraut, R., and Herbsleb, J. 2007. "Team Knowledge and Coordination in Geographically Distributed Software Development," *Journal of Management Information Systems* (24:1), pp. 135–169.
Faraj, S., and Sproull, L. 2000. "Coordinating Expertise in Software Development Teams," *Management Science* (46:12), pp. 1554–1568.
Flanagan, J. C. 1954. "The Critical Incident Technique," *Psychological Bulletin* (51:4), pp. 327–358.
Giddens, A., and Duneier, M. 1999. "Introduction to Sociology," (3rd ed,)New York: W.W. Norton.
Hackman, J. R. R. 1987. "The Design of Work Teams," In *Handbook of Organizational Behavior*, J. Lorsch (ed.), Englewood Cliffs, NJ: Prentice-Hall, pp. 315–342.
Hall, E., and Hall, M. R. 1990. *Understanding Cultural Differences: Keys to Success in West Germany, Francee and the United States*, Intercultural Press, Inc.
Hanssen, G. K., ŠMite, D., and Moe, N. B. 2011. "Signs of Agile Trends in Global Software Engineering Research: A Tertiary Study," In *IEEE International Conference on Global Software Engineering (ICGSE'11)*, pp. 17–23.
Highsmith, J. 2002. *Agile Software Development Ecosystems*, Boston, MA: Addison-Wesley Longman Publishing Co., Inc.
Hole, S., and Moe, N. B. 2008. "A Case Study of Coordination in Distributed Agile Software Development," *Software Process Improvement* (16), pp. 189–200.
Holmstrom, H., Conchúir, E. Ó., Ågerfalk, P. J., and Fitzgerald, B. 2006. "Global Software Development Challenges: A Case Study on Temporal , Geographical and Socio-Cultural Distance," In *IEEE International Conference on Global Software Engineering (ICGSE'06)*.
Holmström, H., Fitzgerald, B., Ågerfalk, P. J., and Conchúir, E. Ó. 2006. "Agile Practices Reduce Distance in Global Software Development," *Information Systems Management* (23:3), pp. 7–18.
Hossain, E. 2008. "Coordinating Mechanisms for Agile Global Software Development," In *IEEE International Conference on Global Software Engineering (ICGSE'08)*, pp. 257–263.
Huang, H., and Trauth, E. M. 2008. "Cultural influences on temporal separation and coordination in globally distributed software development," In *ICIS 2008 Proceedings*. Paper 134.
Hung, Y. T. C., and Nguyen, M. 2008. "The Impact of Cultural Diversity on Global Virtual Team Collaboration A Social Identity Perspective," In *HICSS 2008 Proceedings*.
Iivari, J., and Iivari, N. 2011. "The relationship between organizational culture and the deployment of agile methods," *Information and Software Technology* (53:5), pp. 509–520.
Jalali, S., and Wohlin, C. 2010. "Agile Practices in Global Software Engineering-A Systematic Map," In *IEEE International Conference on Global Software Engineering (ICGSE'10)*, pp. 45–54.
Jarvenpaa, S., and Leidner, D. E. 1999. "Communication and trust in global virtual teams," *Organization Science* (10:6), pp. 791–815.
Joy, W. H., Fang, Y., and Schroeder, A. 2012. "Transactive Memory Systems, Knowledge Integration, and Team Performance in Geographically Dispersed Teams," In *ECIS 2012 Proceedings*, Paper 47.
Komi-Sirviö, S., and Tihinen, M. 2005. "Lessons learned by participants of distributed software development," *Knowledge and Process Management* (12:2), pp. 108–122.
Lee, G., DeLone, W., and Espinosa, J. A. 2006. "Ambidextrous coping strategies in globally distributed software development projects," *Communications of the ACM* (49:10), pp. 35–40.
Lee, G., and Xia, W. 2010. "Toward agile: An integrated analysis of quantitative and qualitative field data on software development agility," *MIS Quarterly* (34:1), pp. 87–114.
Lee, S., and Yong, H.-S. 2009. "Distributed agile: project management in a global environment," *Empirical Software Engineering* (15:2), pp. 204–217.
Leidner, D., and Kayworth, T. 2006. "Review: A review of culture in information systems research: Toward a theory of information technology culture conflict," *MIS Quarterly* (30:2), pp. 357–399.
Lillis, A. M. 1999. "A framework for the analysis of interview data from multiple field research sites," *Accounting & Finance* (39:1), pp. 79–105.
Lu, Y., Xiang, C., Wang, B., and Wang, X. 2011. "What affects information systems development team performance? An exploratory study from the perspective of combined socio-technical theory and coordination theory," *Computers in Human Behavior* (27:2), pp. 811–822.

Majchrzak, A., Chelleppa, R., Cooper, L., and Hars, A. 2005. "Personalizing knowledge delivery services: a conceptual framework," In *New Frontiers in Knowledge Management*, K. C. DeSouza (ed.), New York: Palgrave-Macmillan, pp. 51–75.
Malone, T. W., and Crowston, K. 1994. "The Interdisciplinary Study of Coordination," *ACM Computing Surveys* (26:1), pp. 87–119.
Oshri, I., Van Fenema, P., and Kotlarsky, J. 2008. "Knowledge transfer in globally distributed teams: the role of transactive memory," *Information Systems Journal* (18:6), pp. 593–616.
Paasivaara, M., and Lassenius, C. 2006. "Could Global Software Development Benefit from Agile Methods?," In *IEEE International Conference on Global Software Engineering (ICGSE'06)*, pp. 109–113.
Patton, M. 1990. *Qualitative Evaluation and Research Methods*, Beverly Hills, CA: SAGE Publications Inc.
Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., and Still, J. 2008. "The impact of agile practices on communication in software development," *Empirical Software Engineering* (13:3), pp. 303–337.
Ramesh, B., Cao, L., Mohan, K., and Xu, P. 2006. "Can distributed software development be agile?," *Communications of the ACM* (49:10), pp. 41–46.
Ren, Y., and Argote, L. 2011. "Transactive Memory Systems 1985–2010: An Integrative Framework of Key Dimensions, Antecedents, and Consequences," *The Academy of Management Annals* (5:1), pp. 189–229.
Rico, R., Miriam Sanchez-Manzanares, Gil, F., and Gibson, C. 2008. "Team Implicit Coordination Processes: A Team Knowledge-Based Approach," *Academy of Management Review* (33:1), pp. 163–184.
Sarker, S., and Sahay, S. 2004. "Implications of space and time for distributed work: an interpretive study of US–Norwegian systems development teams," *European Journal of Information Systems* (13:1), pp. 3–20.
Sarker, Saonee, and Sarker, Suprateek. 2009. "Exploring Agility in Distributed Information Systems Development Teams: An Interpretive Study in an Offshoring Context," *Information Systems Research* (20:3), pp. 440–461.
Schein, E. 1985. "Defining organizational culture," In *Organizational Culture and Leadership* San Francisco: Jossey-Bass, pp. 1–22.
Schwaber, K., and Beedle, M. 2002. "Agile Software Development with Scrum," In *Series in Agile software development*Upper Saddle River, NJ: Prentice-Hall.
Shachaf, P. 2008. "Cultural diversity and information and communication technology impacts on global virtual teams: An exploratory study," *Information & Management* (45:2), pp. 131–142.
Šmite, D., Wohlin, C., Gorschek, T., and Feldt, R. 2009. "Empirical evidence in global software engineering: a systematic review," *Empirical Software Engineering* (15:1), pp. 91–118.
Strode, D., Huff, S., and Tretiakov, A. 2009. "The impact of organizational culture on agile method use," In *HICSS 2009 Proceedings*.
Sutherland, J., Viktorov, A., Blount, J., and Puntikov, N. 2007. "Distributed scrum: Agile project management with outsourced development teams," In *HICSS 2007 Proceedings*.
Tanner, M. 2009. "Communication and culture in global software development: The case of Mauritius and South Africa," *Journal of Information, Information Technology, and Organizations* (4).
Thomas, D. M., and Bostrom, R. P. 2010. "Vital signs for virtual teams: an empirically developed trigger model for technology adaptation interventions," *MIS Quarterly* (34:1), pp. 115–142.
Turk, D., France, R., and Rumpe, B. 2002. "Limitations of Agile Software Processes," In *3rd International Conference on eXtreme Programming and Agile Processes in Software Engineering*, pp. 43–46.
Ven, A. H. V. D., Delbecq, A. L., and Koenig, R. 1976. "Determinants of Coordination Modes within Organizations," *American Sociological Review* (41:2), pp. 322–338.
Walsham, G. 2006. "Doing interpretive research," *European Journal of Information Systems* (15:3), pp. 320–330.